OTOT-A3

Name: Erin May Gunawan

Matric. No: A0200765L

GitHub Link: A2-A3

Demo Link: A3

Task A3.1

Install Metrics Server

1. Open your terminal and ensure the cluster is running and all the deployments, service, and ingress are setup

   ```
   # Check if the kind nodes are running
   docker ps -a

   # Check the cluster info
   kubectl cluster-info

   # Check kubernetes resources
   kubectl get <resource>
   ```

2. Install metrics server on cluster

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
   ```

   Edit the deployment configuration file for metric-server to add the flag `--kubelet-insecure-tls` in the `template.spec.containers[].args[]` field.

   ```
   kubectl -n kube-system edit deploy/metrics-server
   ```

   The final `yaml` should look something like this

   ```
   # ...
   template:
     # ...
     spec:
       containers:
         - args:
   ```

```
          - --cert-dir=/tmp
          - --secure-port=4443
          - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
          - --kubelet-use-node-status-port
          - --metric-resolution=15s
          - --kubelet-insecure-tls
# ...
```

3. Check if the metric-server is running

```
# Check if deployed
kubectl -n kube-system get deploy/metrics-server

# sending the request to the API should return a result
# jq is optional and for formatting the output
kubectl get --raw "/apis/metrics.k8s.io/v1beta1/" | jq '.'
```

It should output something similar to this

```
{
"kind": "APIResourceList",
"apiVersion": "v1",
"groupVersion": "metrics.k8s.io/v1beta1",
"resources": [
 {
   "name": "nodes",
   "singularName": "",
   "namespaced": false,
   "kind": "NodeMetrics",
   "verbs": [
     "get",
     "list"
   ]
 },
 {
   "name": "pods",
   "singularName": "",
   "namespaced": true,
   "kind": "PodMetrics",
   "verbs": [
     "get",
     "list"
   ]
 }
]
```

Create Horizontal Pod Autoscaler

1. Create a manifest file for the Horizontal Pod Autoscaler, the content should be the following:

```
# backend-hpa.yaml

apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: backend
  namespace: default
spec:
  metrics:
    - resource:
        name: cpu
        target:
          averageUtilization: 50
          type: Utilization
      type: Resource
  minReplicas: 1
  maxReplicas: 10
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: backend
```

2. Deploy the HPA and verify that it is created correctly

```
# Deploy to cluster
kubectl apply -f k8s/manifests/backend-hpa.yaml

# Check if created
kubectl get hpa backend
```

It should output

```
NAME      REFERENCE           TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
backend   Deployment/backend  0%/50%    1         10        3          27s
```

The `TARGETS` is defined as `<current util> / <target util>` and should both be defined.

If `current util` exceeds the specified `target util`, the target deployment should scale accordingly.

```
# To get more details
kubectl describe hpa
```

It should output

```
HorizontalPodAutoscaler
Name:                                                  backend
Namespace:                                             default
# ...
Reference:                                             Deployment/backend
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  0% (0) / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       3 current / 3 desired
```

The `Reference` and `Metrics` should be defined correctly.

You can also try opening up multiple tabs of the deployment via your browser and check if the deployment's `current utilization` changes and if it scales up.

```
# Watch for changes
kubectl get hpa backend -w
```

Or for stress-testing for multiple requests run the following in your terminal

```
# If using WSL2 to run, ensure the ip address is set to
# Ethernet adapter vEthernet (WSL)'s IP address
# check via running ifconfig on cmd

# Continously sends request, Ctrl+C to stop
while true; do curl <ip address of localhost>:<portNo>/app; done;
```

After some time running the command, the number of `REPLICAS` should increase. After stopping the command and waiting a couple of minutes, the number of `REPLICAS` should decrease.

Create Zone-Aware Deployment

1. Create the deployment

```
# backend-deployment-zone-aware.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-zone-aware
  labels:
    app: backend-zone-aware
spec:
  replicas: 10
  selector:
    matchLabels:
      app: backend-zone-aware
  template:
    metadata:
      labels:
        app: backend-zone-aware
    spec:
      containers:
        - name: backend
          image: erinmayg/nginx-sample
          ports:
            - name: http
              containerPort: 80
          resources:
            limits:
              cpu: 40m
              memory: 100Mi
      topologySpreadConstraints:
        - maxSkew: 1
          topologyKey: topology.kubernetes.io/zone
          whenUnsatisfiable: DoNotSchedule
          labelSelector:
            matchLabels:
              app: backend-zone-aware
```

2. Deploy and verify

```
# Deploy
kubectl apply -f k8s/manifests/backend-deployment-zone-aware.yaml

# check if created
kubectl get po -lapp=backend-zone-aware -owide --sort-by='.spec.nodeName'
```